DOI 10.1007/s10598-015-9280-3
Computational Mathematics and Modeling, Vol. 26, No. 3, July, 2015

II. INFORMATICS

MACHINE LEARNING METHODS IN AUTHENTICATION PROBLEMS USING PASSWORD KEYSTROKE DYNAMICS

V. Yu. Kaganov[1], A. K. Korolev[2], M. N. Krylov[3], I. V. Mashechkin[4], and M. I. Petrovskii[5]

UDC 004.056.53

We examine the problem of user authentication from keystroke dynamics. A new static authentication method that collects information about user keystrokes is described. Its applicability to the authentication problem is substantiated by experiments and the optimal conditions for the implementation of the method are chosen.

Keywords: authentication, static authentication, machine learning, kernel, fuzzy sets, keyboard, behavioral analysis.

1. Relevance

In most modern information systems, one of the main issues, alongside with data storage and processing, is the granting of access permissions. Permissions prevent unauthorized access by outsiders and also differentiate between the access rights of staff members within the organization. The authentication problem, i.e., verifying the legitimacy of the user seeking to access the system, is thus one of the key problems in information systems.

Various methods are available to solve this problem. The most popular technique used in modern information system is password authentication — authentication by a special string of symbols known only to the user who has permission to access the system. In addition to obvious advantages — passwords are simple to implement and their use is very widespread, password authentication suffers from essential shortcomings: the user may forget the code string or reveal it to other users; if the password is too short or too simple, it can be found by direct enumeration or by dictionary-based enumeration. All this casts serious doubts on password use in high-security systems.

Authentication by means of electronic and digital signatures (EDS) is free from the last shortcoming [1]. However, it is impossible to ensure secure storage of closed keys in this way, because they are stored using other access control tools.

For these reasons, many effective security systems rely on biometrics to identify the user. Biometric systems fall in two categories. One category includes systems using features that differ for natural reasons among different individuals: fingerprints, retina scans, voice patterns, body temperature chart, etc. These systems are effective but quite costly because they require special equipment. The second category includes systems that analyze user behavior; they are based on user experience and skills. They do not require any special equipment and are simple to introduce. One of such systems analyzes user keystroke dynamics.

[1]Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, Russia; e-mail: vladhid@gmail.com.
[2]Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, Russia; e-mail: ak@koroandr.me.
[3]Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, Russia; e-mail: sqarert@gmail.com.
[4]Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, Russia; e-mail: mash@cs.msu.su.
[5]Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, Russia; e-mail: michael@cs.msu.su.

Translated from Prikladnaya Matematika i Informatika, No. 46, 2014, pp. 89–102.

 1046-283X/15/2603–0398

In this article, we consider approaches to user authentication with various models of keystroke dynamics observed while the user is entering the password. A user profile model is proposed and the efficiency of intelligent data analysis algorithms for this model is examined.

2. The Problem

In this article, we consider the static authentication problem.

The method works on a given template, word, or any other predefined text. The input data entered by the user (e.g., the password keystrokes) are assembled and compared with previous successful entry attempts. This approach is an extension of the standard login/password authentication method: the system checks not only *what* the user entered, but also *how* it was entered. Let us note some features of static authentication. First, a small volume of input data is involved. Static authentication typically works in conjunction with password authentication, and manual entry of exceedingly long password (more than 100 characters) is virtually excluded.

Another feature is the use of static data. The same password is generally used to enter the system many times and we can select only a small number of features from the password keystrokes. Moreover, the password is changed very seldom, which allows the system to accumulate a large sample. The method should be optimized to recognize the user from a small set of parameters.

Finally, static authentication requires fast processing: the user will not be allowed to work until the authentication has been completed successfully, as this is required to access the system. The delay between entering the password and getting permission to enter the system should therefore be made as small as possible.

More formally, the authentication problem can be described as follows: given is a set of *users* that perform certain *actions*. Pressing a key or entering a password are examples of such actions. In this setting, the learning problem associates to each user a certain function (model) that measures to what extent the user action is anomalous. The authentication problem applies the model to compute the anomaly of the new user action and decides, based on the anomaly measure, if authentication is successful or not.

To assess our results (as well as the results of other researchers), we computed two measures: False Rejection Rate (FRR) and False Acceptance Rate (FAR) [2]. FRR is the percentage of access attempts with a password on which the system has been trained that are perceived as the keystrokes of another user (type I error). FAR is the percentage of access attempts with a strange user password that the system accepts as the keystrokes of the user on which the system has been trained (type II error).

Algorithms (including our proposed algorithm) do not return a binary value (authentication successful/unsuccessful). Instead, they return a real value that shows to what extent the attempt corresponds to training data. It therefore makes sense to introduce some threshold value separating successful and unsuccessful authentication attempts. In what follows, by varying the threshold, we find the value when FRR is equal to FAR. The error with this threshold is called Equal Error Rate (EER) and it is one of the basic indicators for assessing the algorithm performance [2].

The data used in the experiment contain information about password entry by many individuals, and we may accordingly assume that the algorithm results will differ depending on the tested user. A good algorithm should obviously produce equally good results regardless of the person being authenticated. The higher the scatter of EER for various individuals, the more difficult it is to apply the algorithm in practice. Therefore, mean square deviation of EER obtained during authentication of different users is another important parameter for assessing the algorithm performance.

3. Existing Approaches

The existing approaches utilize various features that are collected while the user types and models constructed with these features. We will briefly describe some of these approaches from [3, 4, 5, 6, 7]; a more detailed overview is given in [8].

Key Press Duration. In this approach [4], the model is the vector X of n elements, each element corresponding to one of the keyboard keys and represented by the pair (M_k, D_k): M_k is the mean dwell time during which the key k is depressed (key press duration) and D_k is the standard deviation for key k. This approach has produced FAR and FRR of less than 0.1 simultaneously.

Press-and-Release Sequence [3]. This approach assumes that the user entering the password sometimes presses the next key before releasing the current key (this leads to a so-called "swap"). A user model is constructed by observing the sequence in which keys are pressed and released and calculating the number of "swaps". FAR and FRR produced by this method are highly dependent on pairs of users participating in the assessment, and experimental results produce error rates ranging from 0 to 0.7. This method cannot be regarded as appropriate, other than in combination with other approaches.

Relative Typing Speed. The typing rate is generally assumed the same for any pair of keys, regardless of the text being typed. It is therefore proposed to measure the typing rate for pairs of keys and apply it as a user model. The user model is constructed by measuring the distance between vectors of key pairs ordered by typing speed [5]. The distances between two vectors of the same user were on average 0.3192; the distances between the vectors of different users were 0.529. Authentication can be regarded successful only if the difference between the vectors is less than 0.3 and unsuccessful if the difference is greater than 0.6.

Use of Right and Left Shift Keys. This approach [3] assumes that different people use the right and left `Shift` keys differently, and this may be applied for authentication. Users were divided into 4 classes based on experimental data: those who use only right or only left `Shift`, and those who give preference to the right or the left `Shift`, but also use the other key. If the user falls in the expected class, this still does not mean successful authentication, because there are a total of four classes, and the probability to mistake an impostor for a legitimate user is very high. However, if the user falls in a strange class, this is grounds for rejecting the authentication attempt.

Method for Short Alphabetic or Numeric Passwords. Key press durations (dwell times) are used as a model in [6], where they are calculated by three different techniques. The learning algorithm is the multiclass linear SVM [7], because it demonstrates the best performance on simple data structures. The test subjects were divided into two groups for data collection: one group was aware of the ongoing experiment, the other not. The experiment has shown that participant knowledge affects the results: FAR and FRR in the informed group where a factor of 3 to 5 lower (0.01–0.03).

4. Proposed Approach

4.1. Description of Proposed Model and Algorithms. The feature space for the algorithm is chosen as follows: each attempt to enter the password is a vector of delays between different events of pressing and releasing a key. Starting with the second key depressed, the delays for each key are measured as follows: the time during which the key is pressed, the time between two key press events, and the time between pressing a key and releasing the preceding key. For the first key, only the dwell time is measured. This feature space [9] has produced good results. It was adopted for the proposed approach to allow objective comparison with other algorithms from [9].

As the proposed approach, we consider the static authentication approach [10], which performs quite well in various instances of intrusion detection. In this article, we adapt the method of [10] to our problem.

The proposed approach relies on two main ideas: using the kernel function to calculate distances and using fuzzy set theory to construct the user model. We will consider each of these ideas in more detail.

4.1.1. Kernel Function. The kernel function mechanism [11] is widely used in various machine-learning algorithms (SVM and others). The idea amounts to the following.

Given is some space X and transformation φ that maps from X into a multidimensional (infinite-dimensional) Hilbert space H,

$$\varphi\colon X \longrightarrow H. \tag{1}$$

H is called the *feature space*. The *kernel function* is defined as the scalar product in H:

$$K(x,y) = \langle \varphi(x), \varphi(y) \rangle \tag{2}$$

We can consider the function $K(x,y)$ as a measure of proximity of two elements from X and use it to construct distance functions.

There is no need to evaluate $\varphi(x)$ and $\varphi(y)$ in this case, and the mapping is determined entirely by the kernel $K(x,y)$. This leads to considerable savings of computing time and is one of the main advantages of kernel functions. The use of various kernels imparts a certain flexibility to the proposed approach and broadens the options for choosing a configuration that ensures optimal results.

In the proposed approach, we consider the following distance function based on $K(x,y)$:

$$d(x,y) = \sqrt{K(x,x) - 2K(x,y) + K(y,y)}. \tag{3}$$

Of special interest is the representation of the kernel in the form of the scalar product

$$K(x,y) = \langle x, y \rangle \tag{4}$$

and the Gaussian kernel

$$K(x,y) = \frac{e^{-(x-y)^2}}{2\sigma^2} \tag{5}$$

(in the last case, σ can be chosen so as to achieve the best results).

4.1.2. Fuzzy Clustering in the Feature Space. A fuzzy set [12] is an extension of the classical notion of set. The measure of membership of an element x in a fuzzy set X takes values on $[0,1]$.

In the proposed approach, we seek a fuzzy cluster that contains images of elements from the original space X. As the measure of membership of a fuzzy image in a fuzzy cluster, we may take its measure of correspondence to the cluster, i.e., the reciprocal of anomaly. Images with a small measure of correspondence (less than the threshold established for the particular user) are declared invalid authentication attempts.

It is shown in [10] that the search for a fuzzy cluster in the Hilbert feature space leads to the following fuzzy clustering problem:

$$\begin{gathered} \min_{U\in[0,1]^N,\ a\in H} J(U,a), \\ J(U,a) = \sum_{i=1}^{N} (u_i)^m (\varphi(x_i) - a)^2 - \eta \sum_{i=1}^{N} (1-u_i)^m \end{gathered} \tag{6}$$

Here H is the feature space whose elements are vectors representing the authentication attempts, a is the center of the cluster corresponding to authentication attempts of a legitimate user, N is the number of valid authentication attempts used for training, $u_i \in [0,1]$ is the degree of membership of the image $\varphi(x_i)$ in the cluster

and, as a consequence, the measure of correspondence of the object x_i to the cluster, m is the degree of fuzziness, and η is the distance from the center of the cluster, where the degree of membership is assumed 0.5.

The following iterative algorithm is proposed in [10] for minimizing $J(U, a)$ (it is based on the block-partitioned coordinate descent method):

`Step 0.` Initialize U and η (by simplified choice — see below).

`Step 1.` Find the cluster center:

$$\langle a, a\rangle = \left(\sum_{j=1}^{N} u_j^m \sum_{i=1}^{N} u_i^m K(x_i, x_j)\right) \Bigg/ \left(\sum_{i=1}^{N} u_i^m\right)^2. \tag{7}$$

`Step 2.` For all j from $[1, N]$ calculate the distance to the new cluster center:

$$\langle \varphi(x_j), a\rangle = \left(\sum_{i=1}^{N} u_i^m K(x_i, x_j)\right) \Bigg/ \left(\sum_{i=1}^{N} u_i^m\right). \tag{8}$$

`Step 3.` For all j from $[1, N]$ calculate the new degrees of membership of the training vectors:

$$u_j = \frac{1}{1 + \left(\dfrac{\langle a, a\rangle + K(x_j, x_i) - 2\langle \varphi(x_j), a\rangle}{\eta}\right)^{m-1}}. \tag{9}$$

Repeat `Steps 1--3` until

$$\|U^l - U^{l-1}\| < \varepsilon, \tag{10}$$

where l is the step index, ε is the required accuracy.

The anomaly function $F(x, X)$ takes the form

$$F(x, X) = \frac{1}{1 + \left(\dfrac{\langle a, a\rangle + K(x, x) - 2\langle \varphi(x, X), a\rangle}{\eta}\right)^{m-1}}, \tag{11}$$

where

$$\langle \varphi(x, X), a\rangle = \left(\sum_{i=1}^{N} u_i^m K(x_i, x)\right) \Bigg/ \left(\sum_{i=1}^{N} u_i^m\right), \tag{12}$$

$$u_1, \ldots, u_N \in X, \quad |X| = N.$$

To apply this algorithm, we need to choose the initial values for the elements U and η. As the initial values of U we can take equal values $u_i^0 = 1/N$.

The initial values of η can be chosen in two ways. With simplified choice, η is taken equal to the squared distance between the farthest elements in the training set and does not change during the entire process. Alternatively, the cluster radius is estimated in each iteration by the squared distance from the cluster center to the farthest

vector that is not an outlier. Outliers are defined as the proportion of the vectors farthest from the cluster center (this proportion is a prescribed parameter of the algorithm). In this case, the value of $F(x, X)$ is greater than 0.5 if the vector x is inside the cluster, less than 0.5 if it is outside the cluster, and equal to 0.5 if the vector x is at the cluster boundary. Therefore, in the implementation of the model, 0.5 is taken as the initial value of the minimal degree of typicality for accepting a particular password attempt.

4.1.3. Preprocessing. The collected data may contain heterogeneous features with values falling between various limits. It therefore makes sense to normalize the data for the algorithm, i.e., to reduce the ranges of all feature values to common limits.

The best normalization method for our problem has been chosen experimentally on a standard data set. This is normalization by the absolute deviation. Suppose that the feature p occurs during training in N password attempts. Then for this feature, the normalization factor for the vector x has the form

$$W_p = \sum_{i=1}^{N} \frac{|x_i - \bar{x}|}{N}, \quad x'_p = x_p / W_p, \tag{13}$$

where $\bar{x}$ is the arithmetic mean of the elements of the vector x, and x' is the normalized vector of the features p.

Among other possible normalization factors, we can mention the square root of the absolute deviation, the interquartile range (IQR), and some others, but we will show later that the absolute deviation actually produces the best results of all the available normalizations. We furthermore preprocess the data by replacing each x in the input stream with $\ln(x + C)$, where the parameter C is sufficiently large so that $\ln(x + C)$ is always defined. This technique works because experiments have shown that the random variables describing various features of password keystrokes follow a distribution close to log-normal. Preprocessed input data were obtained by this preprocessing and absolute-deviation normalization.

5. Experiments

We ran a series of experiments to compare the proposed method with existing methods and to choose the optimal parameters of the classification algorithm. For this purpose, the proposed algorithm was implemented in R [13].

5.1. Experimental Data. The data from [9] were used as the experimental input for assessing the algorithm performance. These data were chosen for several reasons: they fit the static authentication context, they had been used in [9] to run other experiments, they constitute a large dataset, and they are representative.

5.2. Experiment Design. To compare the results of the proposed algorithm with the results of other algorithms, the conditions of our experiment must be identical with the conditions employed in the other algorithm. This requires identical data, identical training sample, and identical performance evaluation sample. Since the data were borrowed from [9], we decided to use the experiment design parameters from the same source. The following parameters were used: to create a single user model, 200 **first** attempts to type the password by this user are fed to the training algorithm; to assess the model performance, 200 **last** password typing attempts by this user and 5 **first** attempts for each of the 50 remaining users are fed to the detection algorithm. This in total amounts to 450 typing attempts.

This experiment design closely corresponds to real-life scenarios of authentication by keystroke analysis. The first attempts are used for training, because learning occurs immediately as the password is changed, when the authorized user only starts to develop his password typing characteristics. Detection, on the other hand, uses

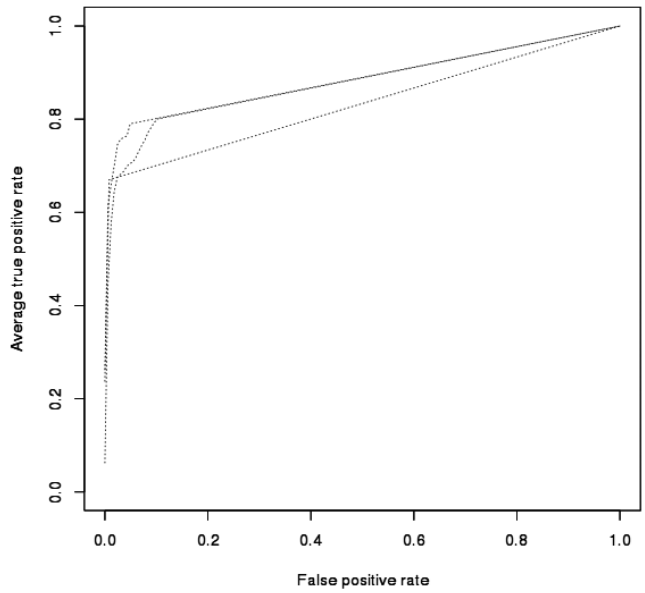

Fig. 1. ROC curves for small values of σ in the Gaussian kernel.

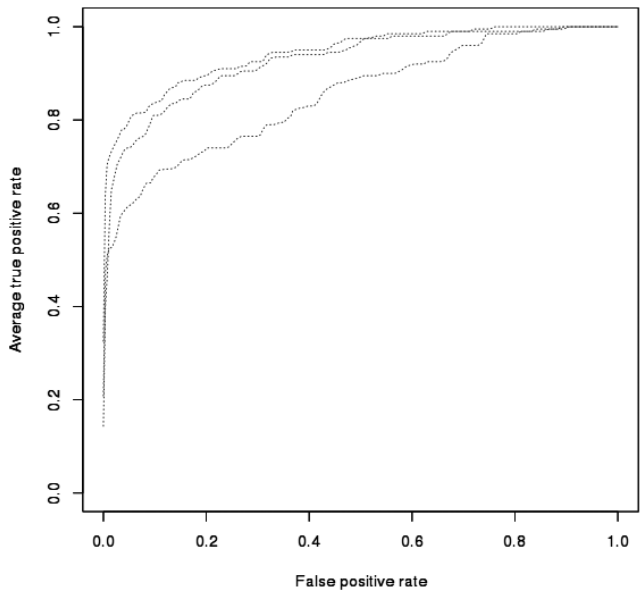

Fig. 2. ROC curves for large values of σ in the Gaussian kernel.

the last attempts, when the authorized user fully displays his password typing characteristics, while unauthorized users, being previously unfamiliar with the password, do not display these characteristics.

5.3. Choice of Parameters and Experimental Results. To assess the effect of the parameters and to choose the best values for our problem, we have carried out a series of experiments varying the parameters between given limits, observing the results, and making appropriate inference about the effect of each parameter on the result. Having selected the best value for one of the parameters, we save this value and continue with the next parameter. In what follows, the parameters are described in the order in which they were chosen.

5.3.1. The Kernel and its Parameters. The scalar product (4) and the Gaussian kernel (5) were used in our experiments.

With the scalar product as the kernel, the best result was EER = 0.31, which implies extremely low accuracy of the algorithm with this kernel function. With the Gaussian kernel, it became necessary to vary also the parameter σ.

For small σ (of the order of 10^{-1}–10^{0}), we observed slow increase of correctly recognized attempts with the decrease of the threshold (Fig. 1). With σ much greater than the optimal value, we observed overall degradation of the ROC curve (Fig. 2) without any characteristic features; with further increase of σ the algorithm failed to converge, probably due to rounding errors in floating point arithmetic. The optimal σ for our sample was $\sigma = 10^{1}$.

5.3.2. Choosing η. The parameter η is the distance from the cluster center, where the degree of membership equals 0.5. This parameter can be chosen by two techniques: simplified and iterative.

With the simplified technique, no additional parameters are required. The best EER obtained by the simplified technique is 0.188. With the iterative technique, we have to specify the proportion of outliers. The optimal expected proportion of outliers was established experimentally (0.1), and with this parameter the iterative algorithm produced EER = 0.178. Varying the proportion of outliers between 0.05 and 0.2, we obtained EER varying between 0.181 and 0.178.

5.3.3. Data Normalization Methods. Essential improvements in the results were obtained when data were normalized after preprocessing. This is so because the parameters by their very nature have widely differing values (thus, for instance, the dwell time can be only positive, while the time between releasing the previous key and pressing the current key is often negative). We accordingly considered several normalization methods for the input data, specifically normalization by the square root of variance, by the absolute deviation, by the square root of the absolute deviation, by the interquartile range (IQR), and by the median absolute deviation.

The best results were obtained with absolute value and square root of absolute value normalizations. The first normalization was better for some users, while the second was better for other users. We tried to elucidate the best normalization for training by performing cross-validation: the training sample was split into two equal parts, the first subsample was normalized by the absolute deviation and the square root of absolute deviation; two models were trained on data with two different normalizations; the models were then tested on the second half of the training sample. In the next stage, training was carried out on the second half with testing on the first half. The best normalization method was the one that produced a lower EER. However, the introduction of cross-validation did not improve the EER.

We furthermore preprocessed the data by replacing each value of x in the input stream with $\ln(x + C)$, as described in Sec. 4.1.3. Combining preprocessing with absolute-deviation normalization, we obtained the final results reported below.

5.4. Experimental Results and Conclusions. We see from Table 1 that the proposed algorithm performs better than the best algorithms from [9]. This proves the applicability and the accuracy of our algorithm for the problem on hand.

Figure 3 shows the ROC curves obtained on ordered data with the proposed algorithms. The thick line is the averaged ROC curve.

6. Conclusion

We have considered static authentication of users from their typing patterns. The basic method uses fuzzy sets and kernel functions, tools that have previously shown good performance for attack detection. A corresponding algorithm has been realized, a series of experiments on standard data have been carried out with the purpose of selecting optimal parameters, and comparison with existing methods has been made. Our algorithm performed

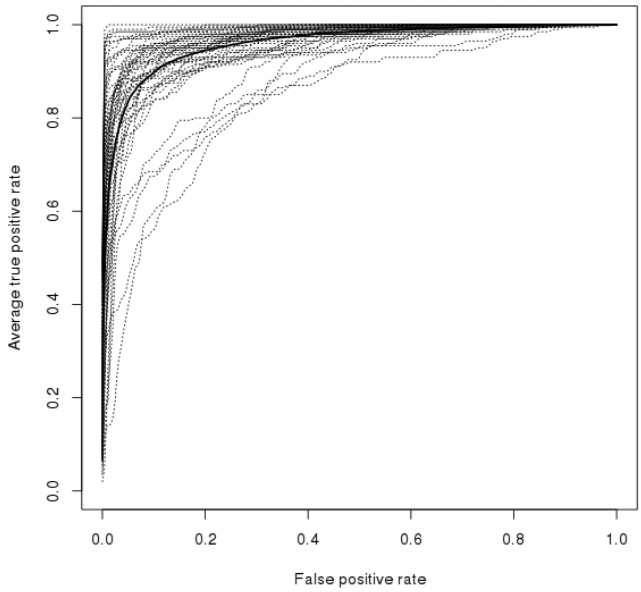

Fig. 3. ROC curves obtained on ordered data using the proposed algorithm.

Table 1. Performance of Various Algorithms

Algorithm	EER	σ
Proposed algorithms	0,092	0,05
Manhattan Scaled[1]	0,096	0,069
Nearest Neighbor	0,100	0,064
Outlier Count	0,102	0,077
SVM	0,102	0,065
Mahalanobis	0,110	0,065
Manhattan Filter	0,136	0,083
Manhattan	0,153	0,092
Neural Network	0,161	0,080

[1] Best of the algorithms in [9].

well in comparison with other algorithms, which proves its applicability to problems of the type considered in this article.

Research supported by the Russian Ministry of Education and Science (agreement 14.604.21.0056).

REFERENCES

1. S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Computing*, **17**, No. 2, 281–308 (1988).
2. M. Krause and H. F. Tipton, *Handbook of Information Security Management*, Auerbach Publications — CRC Press.
3. E. Lau, X. Liu, C. Xiao, and X. Yu, "Enhanced user authentication through keystroke biometrics," *Computer and Network Security, Final project report*, MIT (2004).

4. F. W. M. H. Wong, A. S. M. Supian, and A. F. Ismail, "Enhanced user authentication through typing biometrics with artificial neural networks and k-nearest neighbor algorithm," *IEEE*, 911–915 (2001).
5. F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Trans. Information and System Security*, **5**, No. 4, 367–397 (Nov. 2002).
6. G. Saggio, G. Costantini, and M. Todisco, "Cumulative and ratio time evaluations in keystroke dynamics to improve the password security mechanism," *J. Comput. Inform. Technol.*, **1**, No. 2, 4–11 (2011).
7. K. S. Sung and S. Cho, "GA SVM wrapper ensemble for keystroke dynamics authentication," *Proc. Int. Conf. Biometrics*, ICB, Hong Kong (2004), pp. 654–660.
8. V. Yu. Kaganov, A. V. Korolev, M. N. Krylov, I. V. Mashechkin, and M. I. Petrovskii, "Active authentication by analysis of keystroke dynamics," *Informatika i Ee Primeneniya*, **7**, No. 3 (2013).
9. K. S. Killourhy and R. A. Maxion, *Comparing Anomaly Detectors for Keystroke Dynamics*, IEEE Computer Society Press, Los Alamitos CA (2009).
10. M. I. Petrovsky, "Outlier detection algorithms in data mining systems," *Programming and Computer Software*, **29**, No. 4, 228–237 (2003).
11. B. Scholkopf and A. Smola, *Learning with Kernels* (2002).
12. L. A. Zadeh, "Fuzzy sets," *Information and Control*, **8**, No. 3, 338–353 (1965).
13. R. C. Team, *A Language and Environment for Statistical Computing*, Foundation for Statistical Computing (2005).